FACULDADE UNA DE UBERLÂNDIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOÃO PAULO SOUZA RANIERI

Documentação de um Produto de Software

Strange Adventure

UBERLÂNDIA/MG, 2019

FACULDADE UNA DE UBERLÂNDIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOÃO PAULO SOUZA RANIERI

Documentação de um Produto de Software

Strange Adventure

Coordenadora de Curso: Profª. Valquiria Aparecida Rosa Duarte, Dra.

Orientador(a): Profª. Silvia Fernanda M Brandão, Ms.

Área de Concentração: Sistemas de Informação.
Sub-Área: Desenvolvimento de Jogos

UBERLÂNDIA/MG, 2019

FACULDADE UNA DE UBERLÂNDIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Documentação de um Produto de Software

Strange Adventure

JOÃO PAULO SOUZA RANIERI

Trabalho apresentado à disciplina de Projeto Integrador III, do Curso de Sistemas de Informação, da Faculdade Una de Uberlândia, como parte dos requisitos à graduação e obtenção do título de Bacharel em Sistemas de Informação.

Aprovado em: ___/___/____

Banca Examinadora:

Orientador(a): Profª. Silvia Fernanda M Brandão, Ms.

Co-Orientador(a): Profª. <Nome>, <Titulação (Ms. ou Dr.)>

Examinador(a): Profª. <Nome>, <Titulação (Ms. ou Dr.)>

Examinador(a): Profª. <Nome>, <Titulação (Ms. ou Dr.)>

Examinador(a): Profª. <Nome>, <Titulação (Ms. ou Dr.)>

RESUMO

Este relatório adota como tema o desenvolvimento de jogos voltados para a área de entretenimento; refletindo um interesse particular do autor em desenvolver jogos e, assim, buscar adquirir experiência nessa área, para crescimento profissional e acréscimo de um hobby. O objetivo do software desenvolvido neste relatório está relacionado a um jogo 2D de aventura e segredos, intitulado Strange Adventure. Para isso, foi utilizada a ferramenta gratuita Unity, que permite a criação de cenários, animações, entre outras coisas, além de scripts escritos com C# para controlar todo o ambiente, desde a movimentação do personagem que é jogável até a inteligência artificial dos inimigos que atrapalharão sua aventura. O resultado do estudo apresentado, neste relatório, com o uso da ferramenta Unity foi satisfatório, pois cada problema encontrado é desafiador para aqueles que estão iniciando o desenvolvimento com esta ferramenta, mas cada solução encontrada é gratificante. As expectativas geradas durante o estudo prévio sobre a ferramenta foram atendidos como esperado, pois a ferramenta possui muitos recursos, como por exemplo, a ferramenta para animação, onde é possível animar uma sequência de imagens e gerar uma ação que ela representará, uma boa estratégia é estudar a ferramenta e analisar outros jogos para ter noção de como elaborar uma história e a montagens do cenário. É muito importante definir bem o que deseja-se desenvolver, assim é possível pesquisar e identificar quais componentes utilizar e como utilizar, para que se tenha um desenvolvimento livre de problemas.

Palavras-chave: Desenvolvimento, Documentação, Jogos, Unity, 2D, C#.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CDU	Caso de Uso
RNF	Requisitos não Funcionais

SUMÁRIO

1. Introdução

O projeto de software em desenvolvimento, apresentado neste relatório, trata-se de um jogo 2D produzido com a ferramenta Unity em sua versão gratuita. Esta ferramenta possibilita o desenvolvimento de jogos de vários tipos, como, por exemplo, jogos 3D, de realidade aumentada, e animações, para isso utiliza-se das linguagens C# e C++.

O software, intitulado Strange Adventure, contará com as mecânicas do personagem, que são a movimentação, ataques, interações com objetos e combate com inimigos, e um cenário. Todos os recursos a serem utilizados são disponibilizados pela própria Unity, que possui uma Store onde é possível baixar vários recursos gratuitos, como por exemplo, sprites de personagem, imagens de objetos, fundo, paisagem, terreno. As animações, os cenários, os objetos são todos montados na própria engine.

1.1. Tema e Objetivos do Projeto

Com a evolução da tecnologia, e o avanço dos estudos envolvendo computadores, o homem conseguiu integrar os computadores a dispositivos de vídeo. Um dos primeiros aparelhos de “videogames” utilizava como dispositivo de vídeo um osciloscópio, que é um aparelho utilizado para medição de ondas, ou frequências.

Após o fundamento da Atari(1972), uma das mais importantes empresas do ramo de videogames, esta foi uma das primeiras empresas a comercializar jogos eletrônicos. Contudo, a partir da década de 80 a popularidade dos jogos eletrônicos cresceu muito, e várias empresas foram fundadas, criando diversos tipos de jogos. A maioria era aparelhos arcade, aparelhos que ficavam em estabelecimentos comerciais, com o intuito de obtenção de dinheiro.

A popularidade de jogos 2D cresceu muito nessa década, e vários jogos e aparelhos de videogame foram desenvolvidos, entre os jogos mais populares estavam Super Mario Bros, Pac-Man, Pitfall, Donkey Kong, entre outros.

O projeto, deste relatório, se refere a um jogo de entretenimento, desenvolvido em 2D, utilizando elementos similares ao que eram utilizados nos jogos antigos de plataforma. Este projeto conta com inimigos para dificultar as fases, e obrigações para passar para o próximo nível. Além de explorar o que a ferramenta proporciona, como por exemplo a física, colisão com objetos, utilização de scripts para ações.

O objetivo principal deste relatório é o desenvolvimento de um jogo em 2D. E, dentre os objetivos específicos destacam-se:

- Conhecer e praticar as técnicas de desenvolvimento de jogos com a ferramenta Unity;
- Apresentar as técnicas utilizadas no desenvolvimento de jogo 2D;
- Descrever um plano de testes para o software; assim como a execução e a análise dos resultados.

1.2. Justificativa da Escolha do Tema

A escolha do tema deste projeto se deve ao fato do autor ser um simpatizante por jogos eletrônicos, além do interesse em adquirir conhecimento sobre o mercado de videogames, explorando técnicas de desenvolvimento e ferramentas da atualidade. O autor deseja futuramente trabalhar com desenvolvimento voltado para essa área.

1.3. Descrição Geral do Sistema

O software desenvolvido busca prender a atenção dos jogadores, fazendo com que eles interajam com os cenários e passem por obstáculos, podendo progredir pelas fases cumprindo os desafios até que terminem o jogo.

Os principais afetados pelo sistema serão pessoas de qualquer idade que tenham acesso a um computador e internet para baixar, instalar e usufruir do produto.

1.3.1. Principais Envolvidos e suas Características

Dentre eles destacam-se:

1.3.1.1. Usuários do Sistema

O sistema é livre e será acessível para qualquer pessoal que tenha um computador e acesso à internet, desde crianças a adultos.

1.3.1.2. Desenvolvedores do Sistema

Este projeto foi desenvolvido por João Paulo Souza Ranieri, que realizou o desenvolvimento do código, as animações e a montagem dos cenários. O sprites utilizados foram encontrados na Asset Store da Unity e todos são de uso gratuito.

1.3.2. Sistemas Semelhantes

Como foi utilizado um pacote de assets gratuitos, existem alguns vídeos no YouTube de pessoas utilizando-os para ensinar algo, alguns projetos utilizando alguns seus componentes, e um vídeo do próprio criador com uma demo do que pode ser feito com o pacote.

1.3.3. Restrições e Regras de Negócio

Este projeto não tem finalidade comercial, logo, seu código será disponibilizado na internet para que outras pessoas possam utilizá-lo como referência em estudos e desenvolvimento de jogos similares. No momento será desenvolvido para computadores, sem pretensão de desenvolvimento para plataformas mobile, por mais que a engine utilizada ofereça suporte.

2. Requisitos do Sistema

Esta seção tem como objetivo descrever os requisitos funcionais e não funcionais utilizados no jogo.

2.1. Requisitos Funcionais

A seguir estão listados os requisitos funcionais implantados no projeto. Dentre eles:

- Movimentação.
- Colisão com objetos.
- Interação com objetos.
- Interação com inimigos.
- Sistema de vida e morte.

Os casos de uso que representam as ações executadas dentro do jogo são:

- CDU 1 – Iniciar jogo.
- CDU 2 – Movimentar Personagem.
- CDU 3 – Colisão com Objetos.
- CDU 4 – Interação com Objetos.
- CDU 5 – Interação com Inimigos.
- CDU 6 – Sistema de vida e morte.

As Tabelas de 1 a 6 apresentam os requisitos funcionais e as ações de cada caso de uso associado a eles.

Tabela 1 – Requisito Funcional "Iniciar Jogo"

Nome CDU 1	Iniciar o jogo.
Ator Principal	Jogador
Resumo	Executar o jogo.
Pré-condições	Realizar o download e a instalação do jogo.
Fluxo Principal	1 – O jogador executará o jogo. 2 – O menu do jogo será exibido pelo sistema. 3 – O jogador deverá apertar o botão informado na tela para iniciar o jogo. 4 – O sistema carregará a fase inicial com as instruções do jogo.
Fluxo Alternativo	O jogador encerrará o jogo a partir do menu.

Tabela 2 – Requisito Funcional "Movimentar Personagem"

Nome CDU 2	Movimentar o personagem.
Ator Principal	Jogador.
Resumo	Movimentar o personagem pelo cenário.
Pré-condições	Ter iniciado o jogo a partir do menu inicial.
Fluxo Principal	1 – O jogador utiliza as teclas de movimento de seu teclado, que são representadas com o símbolo de setas. 2 – O personagem obedece aos comandos do jogador, como mover-se para direita e esquerda, e para baixo.
Fluxo Alternativo	1 – O jogador não executa nenhum comando e o personagem fica parado com a animação padrão.

Tabela 3 – Requisito Funcional "Colisão com Objetos"

Nome CDU 3	Colisão com objetos.
Ator Principal	Personagem.
Resumo	Todos os objetos do cenário possuem colisões, inclusive o próprio personagem.
Pré-condições	Ter iniciado o jogo a partir do menu inicial.
Fluxo Principal	1 – Ao iniciar o jogo, o sistema carrega os colisores de cada objeto e do personagem, evitando que ele caia pelo cenário. 2 – O personagem colidirá com o chão. 3 – O jogador poderá movimentar o personagem livremente pelo cenário. 4 – Conforme o jogador avança, aparecerão obstáculos pelo cenário que poderão ser ultrapassados pulando-os ou caindo em buracos.

Tabela 4 – Requisito Funcional "Interação com Objetos"

Nome CDU 4	Interação com objetos.
Ator Principal	Personagem e Jogador.
Resumo	Alguns objetos no cenário poderão ser utilizados pelo jogador.
Pré-condições	O jogador precisa encontrar os objetos e se aproximar.
Fluxo Principal	1 – O jogador se aproxima de um objeto interativo. 2 – Ao aproximar-se, um ícone surgirá na tela informando que aquele objeto é interativo. 3 – O jogador precisará pressionar a tecla de pulo para interagir co o objeto.
Fluxo Alternativo	1 – Caso o objeto não possua interação, o ícone de interação não aparecerá e nada ocorrera além da ação original do botão caso o jogador o pressione.

Tabela 5 – Requisito Funcional "Interação com inimigos"

Nome CDU 5	Interação com inimigos.
Ator Principal	Personagem e Jogador.
Resumo	Todos os inimigos do cenário são interativos.
Pré-condições	O jogador precisa encontrar os inimigos e se aproximar.
Fluxo Principal	1 – O jogador se aproxima de um inimigo. 2 – Ao aproximar-se, o inimigo poderá ou não ver o personagem dependendo da posição que estiver. 3 – Caso o jogador encoste no inimigo, ou vice e versa, o personagem perderá vida. 4 – Caso o jogador pule em um inimigo, o mesmo será morto e sumirá da tela.

Tabela 6 – Requisito Funcional "Sistema de vida e morte"

Nome CDU 6	Sistema de vida e morte.
Ator Principal	Personagem.
Resumo	O personagem possui uma quantidade de vida limitada.
Pré-condições	Iniciar o jogo a partir do menu inicial.
Fluxo Principal	1 – Ao iniciar o jogo o personagem começa com 3 vidas. 2 – As vidas serão perdidas coso o jogador seja acertado por um inimigo ou colida com objetos que causam dano. 3 – Caso as vidas do jogador acabem, o personagem morrerá e a fase será reiniciada.

A seguir serão exibidos nas Figuras de 1 a 6 os casos de uso que representam as ações descritas anteriormente, dentre elas, por exemplo, movimentar personagem.

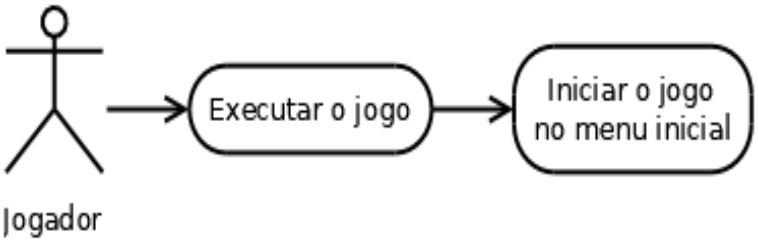

Figura 1: Caso de Uso 1 – Iniciar Jogo

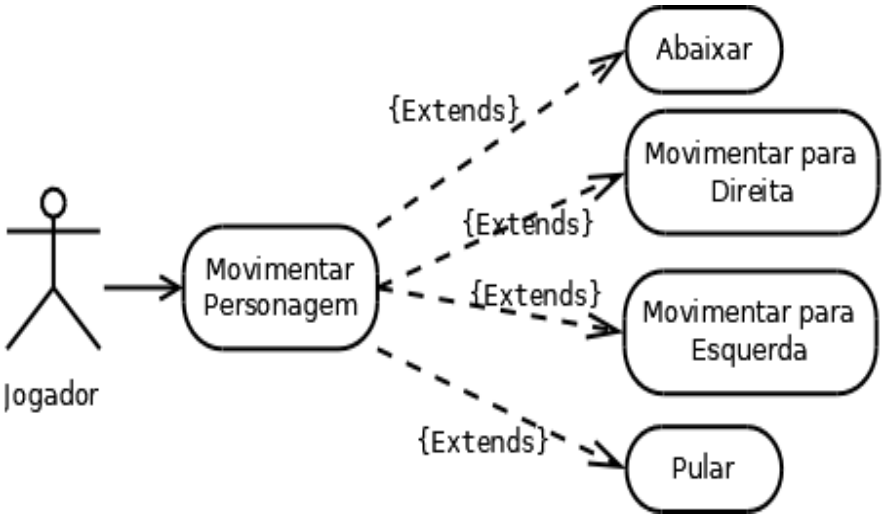

Figura 2: Caso de Uso 2 – Movimentar Personagem

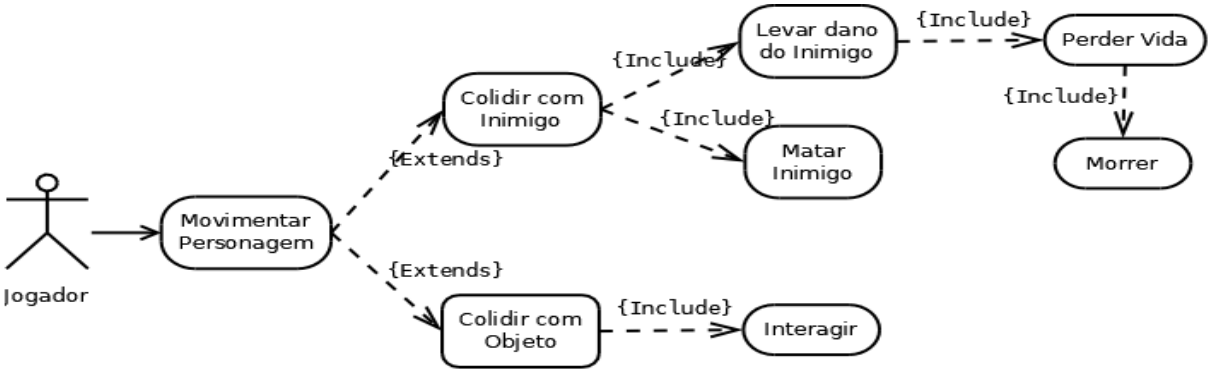

Figura 3: Casos de Uso: 3 – Colisão com objetos, 4 – Interação com objetos, 5 – Interação com inimigos, 6 – Sistema de vida e morte.

2.2. Requisitos Não-Funcionais

Os requisitos não funcionais especificam restrições sobre os serviços ou funções providas pelo sistema. A seguir são apresentados alguns tipos de requisitos não funcionais.

- RNF01 – Consumo de recursos.

– RNF02 – Sistema para Execução.

– RNF03 – Segurança.

As Tabelas de 7 a 9 descrevem as características de cada requisito não funcional, dentre elas consumo de recurso, sistema para execução e segurança.

Tabela 7 – Requisito Não Funcional "Consumo de Recursos"

Indicador:	RNF01	Categoria:	Desempenho
Nome:	Consumo de recursos.		
Data da criação:	24/04/2019	Autor:	João Paulo Souza Ranieri
Data da última alteração:	N/A	Autor:	N/A
Versão:	1	Prioridade:	Essencial
Descrição:	O jogo possui gráficos simples e não precisa de muito processamento tanto do processador quanto da placa de vídeo.		

Tabela 8 – Requisito Não Funcional "Sistema para Execução"

Indicador:	RNF02	Categoria:	Compatibilidade
Nome:	Sistema para Execução		
Data da criação:	24/04/2019	Autor:	João Paulo Souza Ranieri
Data da última alteração:	N/A	Autor:	N/A
Versão:	1	Prioridade:	Essencial
Descrição:	Inicialmente o jogo rodará somente da plataforma Windows, então é necessário que o usuário tenho um computador com este sistema operacional.		

Tabela 9 – Requisito Não Funcional "Segurança"

Indicador:	RNF03	Categoria:	Segurança
Nome:	Segurança		
Data da criação:	24/04/2019	Autor:	João Paulo Souza Ranieri
Data da última alteração:	N/A	Autor:	N/A
Versão:	1	Prioridade:	Essencial
Descrição:	Como o jogo não utiliza nenhum recurso da internet, ele não possui comunicação com o meio externo ao computador que está instalado.		

2.3. Protótipo

Um protótipo consiste em desenvolver uma versão mais básica do software em análise, com o intuito de avaliar a viabilidade e os pontos de sucesso dele antes de entrar em produção. Sendo possível, com o protótipo, identificar pontos a serem melhorados, possíveis falhas e, até planejar uma estratégia que possibilite um desenvolvimento mais rápido, e com baixas chances de erro.

Neste item serão apresentadas algumas imagens do software Strange Adventure, de sua ambientação, personagem e inimigos.

A Figura 4 mostra o local onde a fase se iniciará e o personagem que o jogador utilizará.

Figura 4: Local onde se inicia a fase.

Na Figura 5 tem-se um exemplo de como serão os inimigos, e como poderão ser evitados, bastando utilizar a função de pulo.

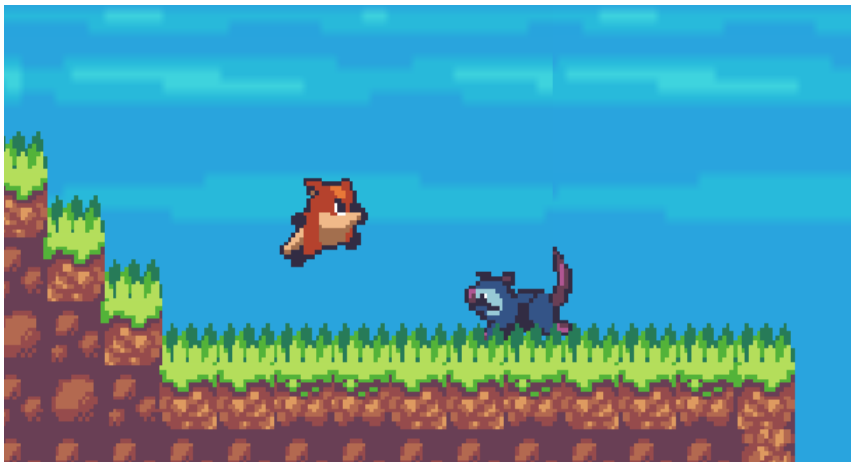

Figura 5: Inimigo e movimento de pulo do personagem.

Na Figura 6 pode-se observar como é realizado o ataque do personagem. Este após pular em cima do inimigo, fará com que o inimigo acabe morrendo. Em seguida é realizada uma animação.

Figura 6: Inimigo executando animação de morte após receber um ataque.

Na figura 7 pode-se observar algumas linhas verdes e uma grade por toda a imagem. As linhas verdes representam os colisores existentes nos objetos exibidos, enquanto a grade é o plano onde é desenhado o cenário.

No personagem existem 3 colisores, um circular e dois em forma de caixa. Logo abaixo observa-se também os colisores do chão que são gerados no formado da grade. Como o chão é formado por emendas de blocos, se for utilizado um colisor em forma de caixa, suas quinas podem travar no chão fazendo com que o personagem trave em alguns pontos. Para evitar este problema é aconselhado utilizar o colisor circular, desta forma seria como se o personagem se deslocasse em cima de rodas, evitando a colisão com as emendas do chão. Ao utilizar qualquer tipo de colisor é importante travar a rotação do personagem no eixo Z, caso contrário ele cairá para o lado movimentado.

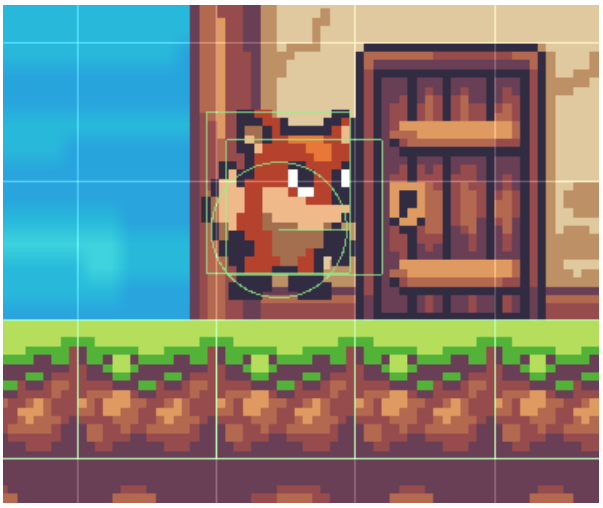

Figura 7: Exemplo de colisores do personagem e do cenário.

Por fim, existem dois colisores de caixa no corpo do personagem, uma mais para baixo e para direita, que será ativado somente quando o personagem executar o movimento de abaixar, e o outro será ativado sempre que ele não estiver abaixado. Deve se utilizar mais de um colisor para os personagens devido à mudança de posição.

Na Figura 8 é mostrado um componente chamado Point Light que encontra-se nas tochas. Este componente é responsável por realizar a iluminação do ambiente, e deve ser posicionado também no eixo Z, distanciando do cenário, como se fosse em um ambiente 3D, pois sua luz é direcionada de seu centro a outro ponto.

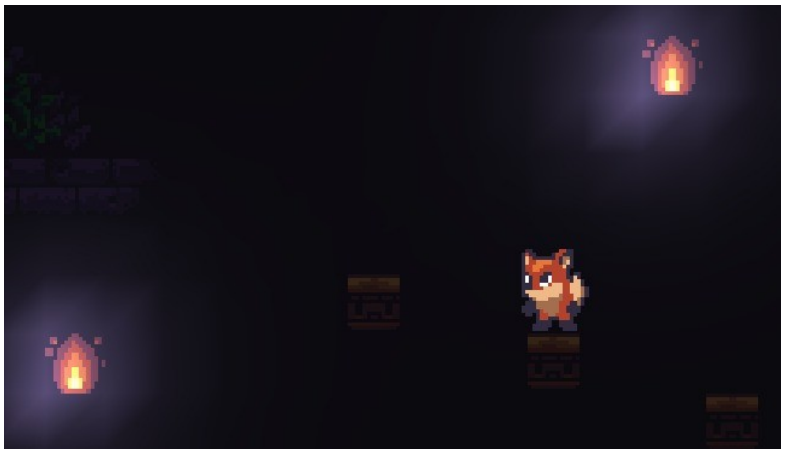

Figura 8: Exemplos de iluminação.

2.3.1. Diagrama de Navegação

A Figura 9 representa a transição das telas do jogo. Assim, o Jogador deverá iniciar o jogo a partir do menu inicial apertando a tecla informada na tela; feito isso, será direcionado para uma tela onde serão exibidos os comandos do jogo e o objetivo das fases. O jogador poderá a qualquer momento retornar para tela de menu do jogo ou encerrá-lo.

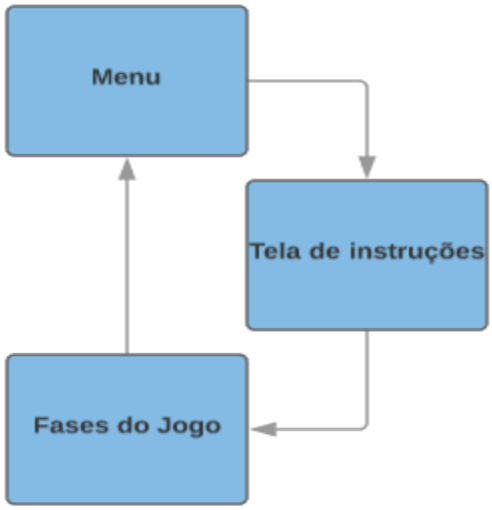

Figura 9: Transição de telas.

2.4. Métricas e Cronograma

Um cronograma é uma ferramenta que permite gerir o desenvolvimento de um projeto, contemplando as atividades realizadas e seu tempo gasto. Sendo assim, a Tabela 10 apresenta os recursos utilizados neste projeto e o tempo gasto em cada uma das atividades, entre outras informações.

Tabela 10 – Cronograma de Desenvolvimento do Software Strange Adventure.

Atividades	Responsável	Plataforma	Tempo
Planejamento	João Paulo	-	1 mês
Aquisição de Software	João Paulo	Unity, Visual Code (Gratuitas)	3 horas
Desenvolvimento	João Paulo	Unity	1 mês
Teste	João Paulo	Unity	20 horas
Documentação	João Paulo	Libre Office	1 mês

3. Análise e Design

Esta seção tem como objetivo analisar e detalhar a solução do sistema de acordo com os requisitos levantados e validados no capítulo 2.

3.1. Arquitetura do Sistema

O sistema desenvolvido é offline e não precisará de comunicação com servidor. A utilização de um servidor será somente para armazenar a página de download do software que terá o link para download e a apresentação do jogo.

Os requisitos para executar o jogo em um computador são:

- Windows 7 ou superior;
- Processador dual core ou superior;
- Espaço em disco de no mínimo 1gb;
- Memória RAM de no mínimo 2gb; e
- Internet somente para download.

3.2. Diagrama de Classes

O objetivo do diagrama de classe é descrever quais tipos de objetos existem no sistema e seus relacionamentos. Ele pode oferecer vários tipos de perspectivas, dependendo do observador e, ainda pode representar conceitos de domínio, foco nos métodos e interfaces principais, além das entidades e relacionamentos.

A Figura 10 apresenta a notação básica do diagrama de classes utilizado no jogo desenvolvido neste projeto.

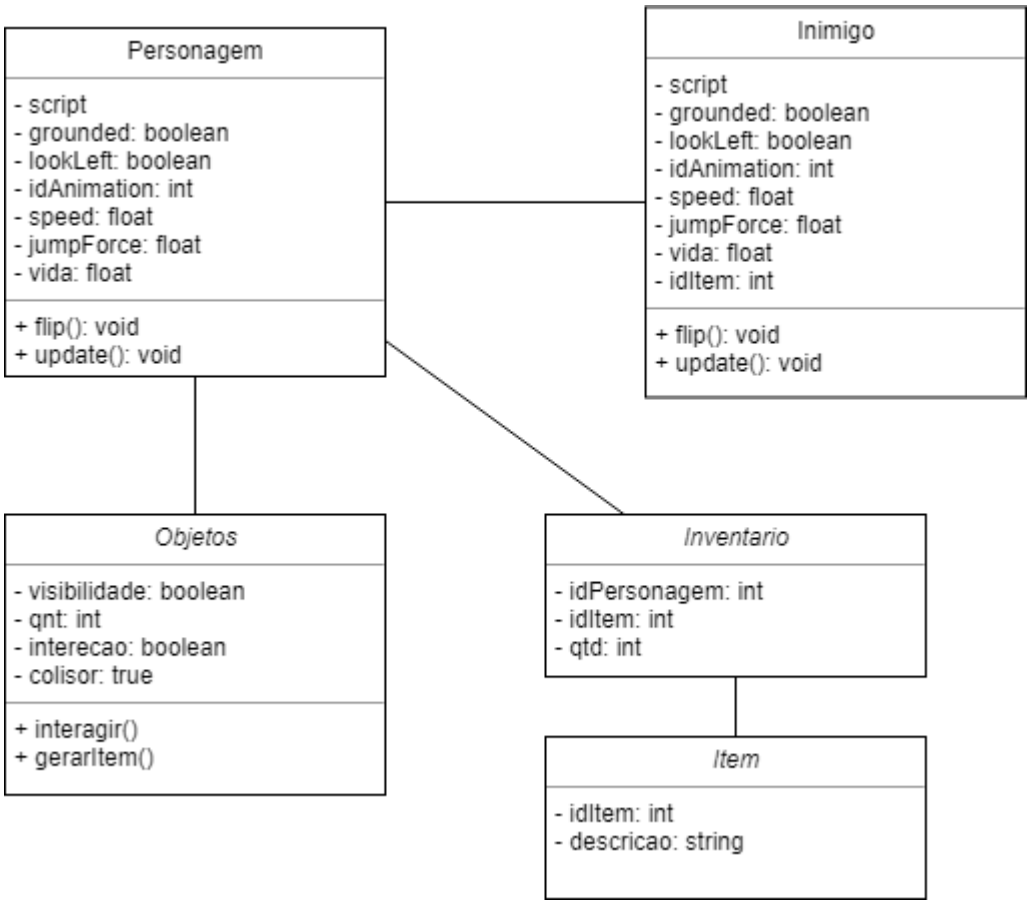

Figura 10: Notação básica do diagrama de classes.

3.3. Modelo de Dados

Como o jogo desenvolvido se trata de um sistema simples e não utiliza muitos recursos que são associados ao personagem, inimigos e usuário, o mesmo não possui banco de dados no momento; pois os recursos necessários são carregados na memória conforme a fase é carregada. Os objetos que precisam ser associados são feitos utilizando vetores com tamanho limitado, armazenando o identificador de cada item.

3.3.1. Modelo Lógico para o Esquema de Dados (DER)

Não se aplica ao software desenvolvido.

3.3.2. Modelo Físico de Banco de Dados (Scripts)

Não se aplica ao software desenvolvido.

3.3.3. Dicionário de Dados

Não se aplica ao software desenvolvido.

3.4. Ambiente de Desenvolvimento

O ambiente de desenvolvimento é tão importante quanto o de produção, onde será disponibilizado o software para download. Como o software não é complexo, e não necessita de muitos recursos, um ambiente de produção com recursos mais simples atenderá perfeitamente, pois disponibilizará somente uma página para download do software, que poderá ser feito por qualquer pessoal.

As Tabelas 11 e 12 apresentam os requisitos mínimos para o ambiente de produção e a configuração do ambiente de desenvolvimento, respectivamente.

Tabela 11 – Requisitos para ambiente de Produção.

Servidor	Linux
Espaço em Disco	20GB
Memória RAM	4GB

Tabela 12 – Configuração do ambiente de Desenvolvimento.

Sistema Operacional	Windows 10
Espaço em Disco	120GB SSD
Memória RAM	8GB
Processador	ADM Ryzen 5 2600

Para o desenvolvimento do software, foi utilizado a Unity, uma engine gratuita (possui versões pagas) que permite a criação de jogos, animações, entre outras coisas. Os códigos necessários para o funcionamento do jogo podem ser escritos com as linguagens C# ou JavaScript, para o software Strange Adventure, foi utilizada a linguagem C#. A Unity não possui um editor de texto nativo, por isso utiliza uma de terceiros, no qual o usuário poderá escolher qual prefere usar. O editor utilizado no desenvolvimento deste projeto foi o Visual Studio 2017 que a própria Unity instalou.

3.5. Sistemas e componentes externos utilizados

Conforme descrito no item 3.4, a ferramenta utilizada para desenvolvimento foi a Unity juntamente do Visual Studio.

A Unity Technologies foi fundada que após o lançamento de um jogo sem sucesso, focaram em disponibilizar para a maior quantidade possível de pessoas a engine que desenvolveram, possibilitando que todos pudessem criar seus jogos 2D e 3D. A Unity se trata de uma ferramenta para criar plataformas 2D e 3D, em tempo real, que possui suporte para diversas plataformas, desde videogames a computadores.

O Visual Studio foi desenvolvido pela Microsoft como um ambiente para desenvolvimento, dedicado ao. NET e as linguagens Visual Basic, C, C++, C# e F#.

4. Implementação

Para uma boa implementação e sucesso no desenvolvimento, é preciso seguir algumas boas práticas durante a escrita do código a fim de que tudo fique padronizado e de fácil entendimento para que qualquer um que leia o código consiga entendê-lo.

Este capítulo tem como objetivo a implementação das classes em termos de componentes, ou seja, toda a implementação deve ser realizada de acordo com as definições das fases anteriores e todos os recursos da programação orientada a objetos que a linguagem escolhida oferece.

A seguir tem-se alguns exemplos de boas práticas que devem ser seguidas, para ter um código bem escrito e descrito. Dentre elas citam-se:

- Comentários no código;
- Padronização de nomes de variáveis, parâmetros, funções, tabelas, *stored procedures*, etc;
- Tratamento de erros;
- Utilização de padrões de projeto;
- Otimização do código, utilizando os melhores algoritmos e funções de recursividade.

A seguir tem-se alguns exemplos do código escrito neste projeto, como mostra a Figura 11 com a definição das variáveis do projeto.

```
private Animator playerAnimator;
private Rigidbody2D playerRb;
public Transform groundCheck; //objeto responsavel por detectar se o personagem esta sobre uma superficie
public Collider2D standing, crouching; //colisor em pe e agachado

public bool Grounded; //indidca se o personagem esta pisando no chao
public bool lookLeft; //indica se o personagem esta virada para esquerda

public int idAnimation; //indica o id da animacao

private float horizontal, vertical;
public float speed; //velocidade de movimento do personagem
public float jumpForce; //forca aplicada para gerar o pulo do personagem
```

Figura 11: Declaração de variáveis.

A figura 12 mostra como é realizada a mudança de direção do personagem e dos inimigos, quando o objeto por completo é virado na direção oposta, evitando-se que os componentes deste objeto saiam da posição. Como por exemplo os colisores que detectam o contato com outros objetos. Imagine como seria desagradável atacar um inimigo com o colisor fora de posição e nada acontecesse.

```
//funcao para virar o personagem
void flip(){
    lookLeft = !lookLeft; //invert o valor da variavel boleana
    float x = transform.localScale.x;
    x *= -1; //invert o valor do scale x
    transform.localScale = new Vector3(x, transform.localScale.y, transform.localScale.z);
}
```

Figura 12: Função responsável por virar o personagem durante a movimentação.

A Figura 13 mostra a função *patrol()* que permite o inimigo movimentar-se pelo cenário patrulhando em busca do jogador. Nela é utilizado o RayCast que é como um laser que informa quando o inimigo entra em colisão com outros objetos; permitindo que o inimigo mude de direção ao colidir com objetos programados, como, por exemplo, uma parede. Isso evitará que ele fique travado na parede.

```
protected void patrol(){
    Debug.DrawRay(transform.position, dir * -1 * distanceChangeRoute, Color.red);
    RaycastHit2D hit = Physics2D.Raycast(transform.position, dir * -1, distanceChangeRoute, layerObstacles);

    if (hit) {
        flip();
        dir *= -1;
    }

    enemyRB.velocity = new Vector2(speed, enemyRB.velocity.y);
}
```

Figura 13: Função responsável por fazer o inimigo patrulhar o cenário.

A figura 14 mostra o Inspector, uma das ferramentas disponíveis no Unity. Ele é responsável por mostrar as propriedades o objeto selecionado, além de facilitar a adição de componentes a ele. Como mostra a figura, existe uma propriedade chamada Transform que é responsável por indicar a posição do objeto no plano 2D ou 3D. O Sprite Rendeder é o responsável por renderizar as imagens do objeto e definir em qual camada ele estará.

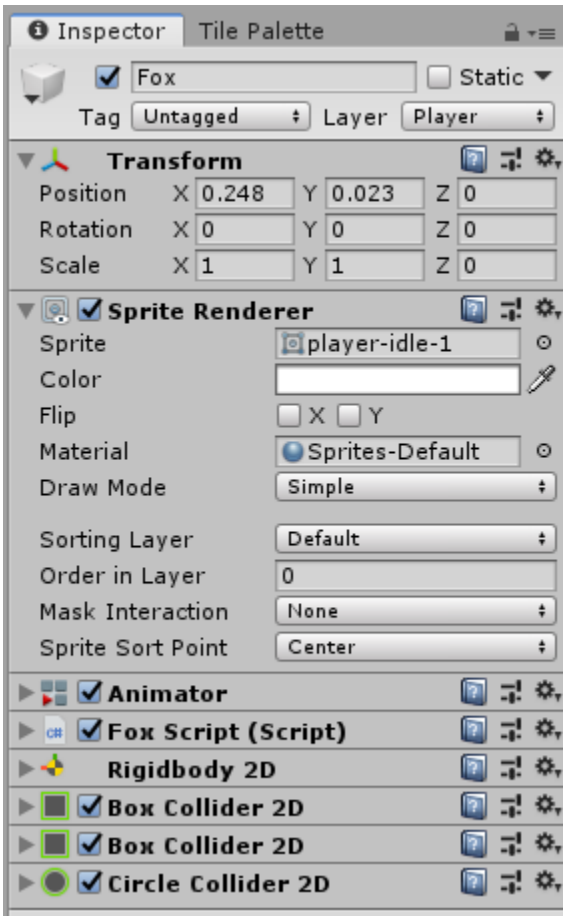

Figura 14: Ferramenta Inspector.

Ainda na Figura 14, tem-se a opção Animator que é responsável por associar o objeto a animação desejada e executá-la. O Script é o responsável por controlar o objeto; e, neste caso o Fox que é o personagem com o qual o jogador controlará, nele são armazenadas todas as informações do player, como vida, funções de movimentar, virar, detectar colisão. O Rigidbody é responsável por definir a massa, colisão, gravidade, entre outras propriedades do objeto. E por fim tem-se os colisores, Box Collider representa os colisores em forma de caixa, e o Circle Collider os colisores em forma de círculo, utilizados na movimentação de objetos.

Na Figura 15 são exibidas algumas abas, as mais importantes são a aba Scene que é onde o trabalho é realizado, como: a montagem do cenário, posição dos objetos, dos inimigos e do jogador. A aba Hierarchy é onde os objetos adicionados na aba Scene ficam listados e agrupados.

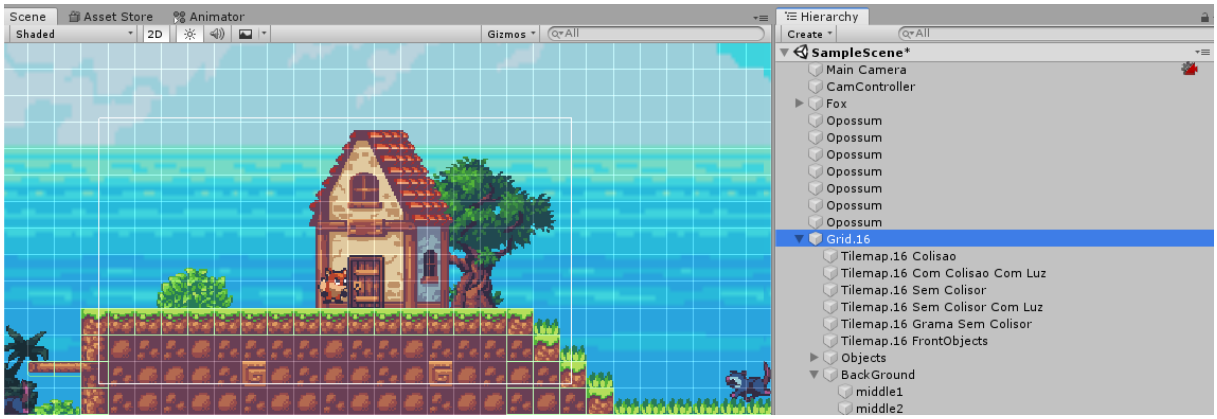

Figura 15: Exemplo de abas da Unity.

Na Figura 16 pode-se observar alguns estados do objeto Fox, o personagem principal do jogo. Nesta ferramenta Animator, é onde se cria a transição de um estado para o outro, definindo a condição que ele ocorrerá.

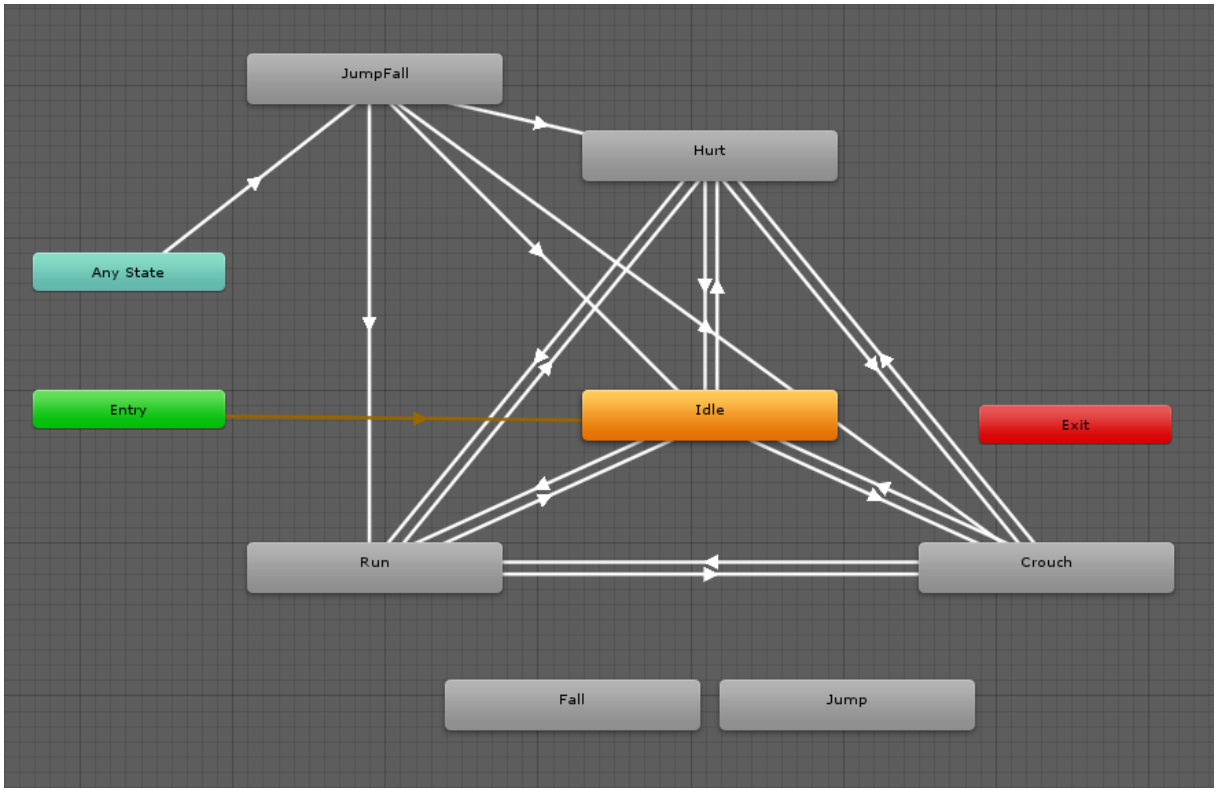

Figura 16: Estados do personagem Fox.

As imagens que foram exibidas e descritas neste capítulo apresentam o ambiente de produção e, partes do desenvolvimento do software; explicando e detalhando um pouco sobre a ferramenta Unity e seu funcionamento, além de algumas dicas sobre o desenvolvimento e a utilização de alguns componentes que são muito úteis.

Alguns dos problemas encontrados durante o desenvolvimento foram expostos no decorrer do capítulo junto de suas correções, realizar pesquisas sobre os componentes que serão utilizados é muito importante, pois auxilia a solucionar algum problema ou bug que pode ocorrer, como por exemplo a utilização do colisor de círculo para movimento em vez do colisor de caixa.

5. Testes

Este capítulo tem como objetivo identificar defeitos no sistema e validar suas funções; verificando se os requisitos foram implementados de forma adequada e ainda avaliar a qualidade do software.

5.1. Plano de Testes

O plano de testes, apresentado na Tabela 13, foi elaborado com o objetivo de validar todas as funcionalidades propostas nos requisitos do projeto, a fim de detectar problemas que poderão comprometer o desenvolvimento do jogo, assim como sua utilização.

Tabela 13 - Plano de testes.

Nº do Teste	Tipo do Teste	Objetivo do Teste	Resultado Esperado
RF0001	Funcional	Validar movimentação do personagem.	Movimentar o personagem livremente conforme os comandos do teclado.
RF0002	Funcional	Validar colisão do personagem e dos inimigos com outros objetos.	Colidir com obstáculos.
RF0003	Funcional	Validar eventos de morte.	Executar a animação de morte quando for preciso.
RF0003	Funcional	Validar interação com outros objetos.	Interagir e executar a ação programada para aquele objeto.

5.2. Execução do Plano de Testes

Neste item serão registrados os testes realizados no desenvolvimento do sistema Strange Adventure, tendo como base o Plano de Testes do Sistema.

Na Tabela 14 observa-se a execução do plano de testes que foi elaborado para este projeto. Como trata-se de um jogo, existem várias validações que precisam ser feitas, como

por exemplo, se a movimentação do personagem está funcionando corretamente, e, se os objetos programados para colidirem entre si, colidirão.

Tabela 14 – Execução do plano de testes.

Nº do Teste	Objetivo do Teste	Resultado Obtido	Comentários
RF0001	Validar a movimentação do personagem.	O personagem realizou a movimentação normalmente pelo cenário após as devidas correções.	Foi necessário definir algumas propriedades de posição para evitar que o personagem travasse no chão durante o deslocamento e não girasse em torno de seu eixo.
RF0002	Validar colisão do personagem e dos inimigos com outros objetos.	Quando um objeto que se movimenta entra em contato com um objeto que não se movimenta, ambos ficam parados. Se dois objetos que se movimentam colidem, o mais forte empurra o outro.	No caso de colisões entre o personagem e os inimigos, ocorrerá uma ação dependendo de onde ocorrer a colisão.
RF0003	Validar eventos de morte.	Quando o colisor do personagem detecta uma colisão com um objeto programado, ele executa uma animação de morte e desaparece, o mesmo se aplica ao inimigo.	O personagem pode receber a colisão de qualquer lado, já o inimigo, só recebe a colisão na parte superior do corpo.
RF0004	Validar interação com outros objetos.	Quando o personagem se aproxima de um objeto que possui interação, é exibido um ícone e ao apertar o botão de ação, a interação será executada.	

5.3 Análise dos Resultados

Os testes executados durante o desenvolvimento do software foram importantes, pois permitiu identificar falhas que afetariam diretamente na execução do jogo e que atrapalhariam quem estivesse utilizando.

Um exemplo dessas falhas é a de que o personagem estava tropeçando na divisão entre cada bloco do terreno, outra foi a colisão entre os inimigos que prejudicava seu movimento, fazendo com que ambos se empurrem.

A elaboração e execução de um plano de testes é essencial para que o produto final atinja a qualidade desejada, e que o usuário tenha a melhor experiência possível.

6. Implantação

Este capítulo visa apresentar informações relevantes para a implantação e funcionamento do sistema.

6.1. Diagrama de Implantação

Neste item tem-se o diagrama básico de implantação do projeto, Figura 17, que representa a parte física do sistema, exibindo o servidor que armazenará o sistema e o computador do cliente que realizará o download e executará o software.

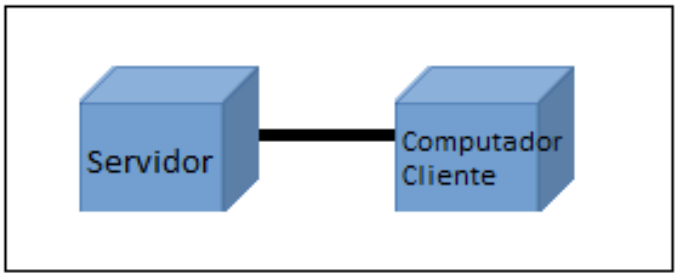

Figura 17: Notação básica do diagrama de implantação.

O software desenvolvido necessita somente de um servidor para hospedá-lo, pois será a partir dele que o cliente efetuará o download. Após realizar o download, o cliente instalará o software da forma convencional, avançando no wizard de instalação padrão do windows. Após finalizada a instalação, o software necessitará somente dos recursos locais do computador do cliente, sem a necessidade de comunicar-se com o servidor.

6.2. Manual de Implantação

Para realizar a implantação do software é necessário seguir as instruções listadas a seguir:

1. A pessoal responsável deverá realizar a configuração de um ambiente de produção, ou seja, um servidor local que tenha uma boa conexão com a internet ou um servidor na nuvem.
2. Configurar o serviço que fornecerá o download do instalador do software, como por exemplo, um site que contenha várias informações e instruções sobre o software.
3. Disponibilizar o ambiente de download para que qualquer um possa acessá-lo.
4. Como a instalação do software não possui nenhuma configuração prévia, não há necessidade de elaborar um manual de instalação para o usuário, pois a Unity permite montar o software para instalação ou montar o executável que abrirá diretamente o jogo. O manual com as informações básicas do jogo será exibido na tela antes de iniciar-se a primeira fase do jogo.

7. Manual do Usuário

Como o software desenvolvido se trata de um jogo, não é necessário realizar a elaboração de um manual, pois os comandos e as instruções de como o ambiente funciona serão informadas durante sua execução, como por exemplo as teclas para se movimentar, para interagir, e como funciona o combate com inimigos.

A Figura 18 mostra o que o usuário poderá realizar no jogo, como a movimentação do personagem que é executada utilizando as teclas direcionais do teclado, o pulo que é executado com a barra de espaço, como eliminar os inimigos, e uma dica sobre como e quem dará as dicas para avançar para próxima fase.

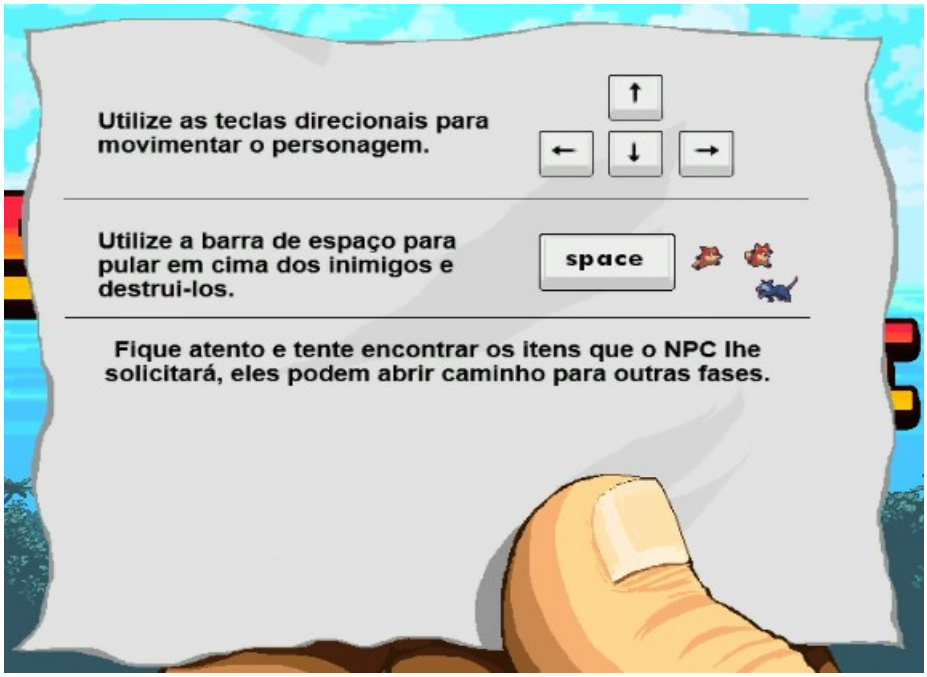

Figura 18: Representa as instruções sobre como o jogo funciona.

8. Conclusões e Considerações Finais

A partir do desenvolvimento do projeto Strange Adventure, foram adquiridas experiências com o ambiente de desenvolvimento Unity, além de mais conhecimento em C#, como também uma visão mais bem relacionada ao desenvolvimento de cenários, histórias e animações.

Os testes realizados também foram de grande importância, pois existem diversos problemas com a combinação de alguns componentes, como por exemplo o colisor de caixa para movimentar o personagem e o colisor de caixa do chão, suas quinas podem colidir e prejudicar a movimentação do personagem. Outro problema é a validação das colisões para executar a animação de morte, é importante detectar qual objeto está colidindo com qual e qual componente de colisão, como por exemplo, o objeto “Inimigo” e seu colisor de caixa colidiram com o objeto “Player” e seu colisor de caixa, nesse caso o player executará uma animação de morte.

A ferramenta Unity possui muitas funcionalidades, a maior parte delas podem ser utilizadas por qualquer desenvolvedor, já que fornece uma praticidade durante a elaboração do jogo. Um exemplo disso é o Animator, que é responsável por executar em um tempo determinado um conjunto de imagem que representarão uma ação, como correr, ficar parado, pular. Os componentes de colisão são essenciais para que os objetos animados não caiam do cenário, e possibilita que se programe uma ação quando um objeto colide com outro.

O conhecimento adquirido servirá para projetos futuros na área de desenvolvimento de jogos, começando com as plataformas 2D e progredindo para o 3D. Com novos temas em mente, a finalização do Strange Adventure servirá de base para que essas ideias saiam do papel e se tornem projetos que serão finalizados e publicados em plataformas que possibilitam o

download de jogos. O desenvolvimento mobile é uma alternativa que futuramente será utilizada, mas no momento o jogo Strange Adventure tem somente uma versão desktop.

Referências Bibliográficas

AMOROSO, Danilo. **A história dos Videogames Games: Do osciloscópio aos Gráficos 3D.** Disponível em: https://www.tecmundo.com.br/xbox-360/3236-a-historia-dos-video-games-do-osciloscopio-aos-graficos-3d.htm. Acessado em: 30 de Março de 2019.

História do Videogame. Disponível em: https://www.historiadetudo.com/videogame. Acessado em: 30 de Março de 2019.

MICROSOFT. Disponível em: https://visualstudio.microsoft.com/pt-br/?rr=https%3A%2F%2Fwww.google.com%2F. Acessado em: 13 de Abril de 2019.

Unity. Disponível em: https://unity.com/pt . Acessado em: 21 de Março de 2019.

Unity Asset Store. Disponível em: https://assetstore.unity.com/?_ga=2.89442331.1440210711.1559235970-871820549.1559235970 . Acessado em: 21 de Março de 2019.

Atari. Disponivel em : https://www.atari.com/about-us/. Acessado em: 22 de Junho de 2019.